

# Struktury danych

## Projekt 1

### *Listy wiązane a tablice dynamiczne*

Mateusz Hypta 280116

Grupa nr.8

Środa 9:15

## Spis treści

1. Wstęp .....	2
2. Opis implementacji .....	2
2.1. Tablica dynamiczna .....	2
2.2. Lista wiązana .....	2
3. Porównanie operacji .....	2
3.1. Dodawanie elementu .....	2
3.2. Usuwanie elementu .....	2
3.3. Wyszukiwanie zadanego elementu.....	2
4. Wyniki i analiza .....	3
5. Porównanie wydajności .....	3
6. Podsumowanie.....	4

## 1. Wstęp

Celem zadania było zaimplementowanie wybranych wariantów listy wiązanej oraz tablicy dynamicznej w języku C++, a następnie porównanie wydajności operacji takich jak dodawanie, usuwanie i wyszukiwanie elementów.

## 2. Opis implementacji

### 2.1. Tablica dynamiczna

Tablica dynamiczna została zaimplementowana jako klasa `DynamicArray`. Klasa ta posiada metody do dodawania elementów na początek, koniec oraz na wybrany indeks, a także metody do usuwania elementów z tych miejsc. W przypadku braku miejsca w tablicy, jej rozmiar jest podwajany.

### 2.2. Lista wiązana

Lista wiązana została zaimplementowana jako klasa `LinkedList`, która korzysta z węzłów (`Node`). Klasa ta posiada metody do dodawania i usuwania elementów na początek, koniec oraz na wybrany indeks, a także metodę do wyszukiwania elementów.

## 3. Porównanie operacji

### 3.1. Dodawanie elementu

**Na początek struktury:** `addToStart`

**Na koniec struktury:** `addToEnd`

**Na wybrany indeks:** `addAtIndex`

### 3.2. Usuwanie elementu

**Z początku struktury:** `removeFromStart`

**Z końca struktury:** `removeFromEnd`

**Z wybranego indeksu:** `removeAtIndex`

### 3.3. Wyszukiwanie zadanego elementu

**search**

## 4. Wyniki i analiza

Testy zostały przeprowadzone dla 100,000 powtórzeń każdej operacji. Oto wyniki:

```
DynamicArray - Average time addToEnd: 9 ns
DynamicArray - Average time addToStart: 119439 ns
DynamicArray - Average time addAtIndex: 316380 ns
DynamicArray - Average time removeFromStart: 583179 ns
DynamicArray - Average time removeFromEnd: 10 ns
DynamicArray - Average time removeAtIndex: 607272 ns
DynamicArray - Search time: 0 ns
DynamicArray - Found index: 777

LinkedList - Average time addToEnd: 247221 ns
LinkedList - Average time addToStart: 49 ns
LinkedList - Average time addAtIndex: 50 ns
LinkedList - Average time removeFromStart: 45 ns
LinkedList - Average time removeFromEnd: 1422087 ns
LinkedList - Average time removeAtIndex: 27 ns
LinkedList - Search time: 0 ns
LinkedList - Found index: 777

Process returned 0 (0x0)   execution time : 848.850 s
Press any key to continue.
|
```

## 5. Porównanie wydajności

Porównując wyniki, można zauważyć, że:

- Dodawanie elementów:
  - Tablica dynamiczna jest szybsza w dodawaniu elementów na koniec struktury.
  - Lista wiązana jest szybsza w dodawaniu elementów na początek struktury
  - Lista wiązana jest szybsza w dodawaniu na wybrany indeks.
- Usuwanie elementów:
  - Lista wiązana jest szybsza w usuwaniu elementów z początku struktury
  - Lista wiązana jest szybsza w usuwaniu elementów z wybranego indeksu.
  - Tablica dynamiczna jest szybsza w usuwaniu elementów z końca struktury.
- Wyszukiwanie elementów:

- Obie struktury mają podobny czas wyszukiwania, który jest bardzo szybki.

## **6. Podsumowanie**

Wyniki pokazują, że tablica dynamiczna jest bardziej wydajna w operacjach dodawania i usuwania na końcu struktury, podczas gdy lista wiązana lepiej radzi sobie z operacjami na początku struktury. Jeśli chodzi o wyszukiwanie elementów, obie struktury są w tej operacji bardzo szybkie. Można powiedzieć, że jedna z tych struktur jest całkowitym przeciwieństwem drugiej, tzn. jeśli wyniki w jednej strukturze są wolne, w drugiej będą szybkie i na odwrót.